**Attention Is All You Need**

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,

Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

**Abstract**

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 Englishto-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

지배적 시퀀스 변환 모델은 인코더와 디코더를 포함하는 복잡한 순환 신경망 또는 합성곱 신경망을 기반으로 합니다. 최고 성능의 모델은 또한 주의 메커니즘을 통해 인코더와 디코더를 연결합니다. 우리는 주의 메커니즘에만 기반을 둔 새로운 단순 네트워크 아키텍처인 Transformer를 제안하고, 반복 및 회선을 완전히 제거합니다. 두 가지 기계 번역 작업에 대한 실험은 이러한 모델이 더 병렬화 가능하고 훈련 시간이 훨씬 적으면서도 품질이 우수함을 보여줍니다. 우리 모델은 WMT 2014 영어-독일어 번역 작업에서 28.4 BLEU를 달성하여 앙상블을 포함한 기존 최고의 결과보다 2 BLEU 이상 향상되었습니다. WMT 2014 영어-프랑스어 번역 작업에서 우리 모델은 8개의 GPU에서 3.5일 동안 교육한 후 41.8의 새로운 단일 모델 최신 BLEU 점수를 설정합니다. 이는 최고 교육 비용의 작은 부분입니다. 문학에서 모델. 우리는 Transformer가 대규모 및 제한된 교육 데이터 모두를 사용하여 영어 구성 요소 구문 분석에 성공적으로 적용함으로써 다른 작업에 잘 일반화됨을 보여줍니다.

**1. Introduction**

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures.

순환 신경망, 특히 장단기 기억[13] 및 개폐 순환[7] 신경망은 언어 모델링 및 기계 번역과 같은 시퀀스 모델링 및 변환 문제에서 최첨단 접근 방식으로 확고히 확립되었습니다. 그 이후로 반복되는 언어 모델과 인코더-디코더 아키텍처의 경계를 넓히기 위한 수많은 노력이 계속되었습니다.

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states , as a function of the previous hidden state and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

순환 모델은 일반적으로 입력 및 출력 시퀀스의 기호 위치를 따라 계산을 고려합니다. 계산 시간의 단계에 위치를 정렬하면 이전 은닉 상태 및 위치 t에 대한 입력의 함수로 은닉 상태 의 시퀀스가 ​​생성됩니다. 이 본질적으로 순차적인 특성은 메모리 제약이 예제 간의 일괄 처리를 제한하기 때문에 더 긴 시퀀스 길이에서 중요해지는 훈련 예제 내 병렬화를 배제합니다. 최근 연구에서는 인수분해 트릭[21]과 조건부 계산[32]을 통해 계산 효율성이 크게 향상되었으며 후자의 경우 모델 성능도 향상되었습니다. 그러나 순차 계산의 근본적인 제약은 남아 있습니다.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.주의 메커니즘은 다양한 작업에서 강력한 시퀀스 모델링 및 변환 모델의 필수 부분이 되어 입력 또는 출력 시퀀스에서 거리에 관계없이 종속성을 모델링할 수 있습니다. 그러나 몇몇 경우[27]를 제외하고는 이러한 주의 메커니즘이 순환 네트워크와 함께 사용됩니다.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

이 작업에서 우리는 반복을 피하고 대신 입력과 출력 사이의 전역 종속성을 끌어내기 위해 주의 메커니즘에 전적으로 의존하는 모델 아키텍처인 Transformer를 제안합니다. Transformer는 훨씬 더 많은 병렬화를 허용하고 8개의 P100 GPU에서 12시간 동안 교육을 받은 후 번역 품질의 새로운 상태에 도달할 수 있습니다.

**2. Background**

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

순차 계산을 줄이는 목표는 또한 확장 신경망 GPU[16], ByteNet[18] 및 ConvS2S[9]의 기초를 형성하며, 모두 기본 빌딩 블록으로 컨볼루션 신경망을 사용하고 모든 입력에 대해 병렬로 숨겨진 표현을 계산합니다. 출력 위치. 이러한 모델에서 두 개의 임의 입력 또는 출력 위치의 신호를 연결하는 데 필요한 작업 수는 위치 사이의 거리에서 증가합니다. ConvS2S의 경우 선형으로, ByteNet의 경우 대수적으로 증가합니다. 이것은 원거리 위치 간의 종속성을 학습하는 것을 더 어렵게 만듭니다[12]. Transformer에서 이것은 평균적인 Attention-Weighted 위치로 인해 감소된 유효 해상도를 희생하더라도 일정한 수의 작업으로 감소합니다. 섹션 3.2에 설명된 대로 Multi-Head Attention으로 상쇄하는 효과입니다.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

때때로 인트라 어텐션(intra-attention)이라고 하는 셀프 어텐션은 시퀀스의 표현을 계산하기 위해 단일 시퀀스의 서로 다른 위치를 연결하는 어텐션 메커니즘입니다. Self-attention은 독해, 추상적 요약, 텍스트 수반 및 학습과제 독립적인 문장 표현을 포함한 다양한 과제에서 성공적으로 사용되었습니다.

End-to-end memory networks are based on a recurrent attention mechanism instead of sequencealigned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

종단 간 메모리 네트워크는 순차 정렬 반복 대신 반복 주의 메커니즘을 기반으로 하며 간단한 언어 질문 응답 및 언어 모델링 작업에서 잘 수행되는 것으로 나타났습니다[34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequencealigned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

그러나 우리가 아는 한, Transformer는 sequencealigned RNN이나 convolution을 사용하지 않고 입력 및 출력의 표현을 계산하기 위해 전적으로 self-attention에 의존하는 최초의 변환 모델입니다. 다음 섹션에서는 Transformer에 대해 설명하고 자기 주의를 환기시키며 [17, 18] 및 [9]와 같은 모델에 비해 이점에 대해 논의합니다.

**3. Model Architecture**

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations to a sequence of continuous representations . Given z, the decoder then generates an output sequence ) of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

대부분의 경쟁적인 신경 시퀀스 변환 모델은 인코더-디코더 구조를 가지고 있습니다[5, 2, 35]. 여기에서 인코더는 기호 표현의 입력 시퀀스를 연속 표현 의 시퀀스에 매핑합니다. z가 주어지면 디코더는 한 번에 한 요소씩 심볼의 출력 시퀀스)를 생성합니다. 각 단계에서 모델은 자동 회귀적이며 [10] 다음을 생성할 때 이전에 생성된 기호를 추가 입력으로 사용합니다. Transformer는 각각 그림 1의 왼쪽과 오른쪽 절반에 표시된 인코더와 디코더 모두에 대해 스택형 self-attention 및 point-wise, fully connected 레이어를 사용하여 이 전체 아키텍처를 따릅니다.

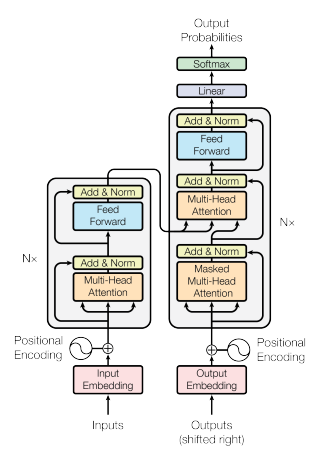


Figure 1: The Transformer - model architecture.

그림 1: 트랜스포머 - 모델 아키텍처.

**3.1 Encoder and Decoder Stacks**

Encoder: The encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, positionwise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is , where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension .인코더: 인코더는 N = 6개의 동일한 레이어 스택으로 구성됩니다. 각 레이어에는 두 개의 하위 레이어가 있습니다. 첫 번째는 다중 헤드 자가 주의 메커니즘이고 두 번째는 단순하고 위치별로 완전히 연결된 피드포워드 네트워크입니다. 우리는 두 하위 계층 각각 주위에 잔여 연결[11]을 사용하고 계층 정규화[1]를 수행합니다. 즉, 각 하위 계층의 출력은 LayerNorm(x+Sublayer(x))이며, 여기서 Sublayer(x)는 하위 계층 자체에서 구현하는 기능입니다. 이러한 잔여 연결을 용이하게 하기 위해 모델의 모든 하위 계층과 임베딩 계층은 차원의 출력을 생성합니다.

Decoder: The decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position can depend only on the known outputs at positions less than .

디코더: 디코더도 N = 6개의 동일한 레이어 스택으로 구성됩니다. 각 인코더 계층의 두 하위 계층 외에도 디코더는 인코더 스택의 출력에 대해 다중 헤드 주의를 수행하는 세 번째 하위 계층을 삽입합니다. 인코더와 유사하게 각 하위 계층 주위에 잔여 연결을 사용하고 계층 정규화를 수행합니다. 또한 디코더 스택의 self-attention 하위 계층을 수정하여 위치가 후속 위치에 주의를 기울이지 않도록 합니다. 출력 임베딩이 한 위치만큼 오프셋된다는 사실과 결합된 이 마스킹은 위치 i에 대한 예측이 i보다 작은 위치에서 알려진 출력에만 의존할 수 있도록 합니다.

**3.2 Attention**

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

주의 기능은 쿼리 및 키-값 쌍 집합을 출력에 매핑하는 것으로 설명할 수 있습니다. 여기서 쿼리, 키, 값 및 출력은 모두 벡터입니다. 출력은 값의 가중치 합계로 계산되며, 여기서 각 값에 할당된 가중치는 해당 키와 쿼리의 호환성 함수에 의해 계산됩니다.

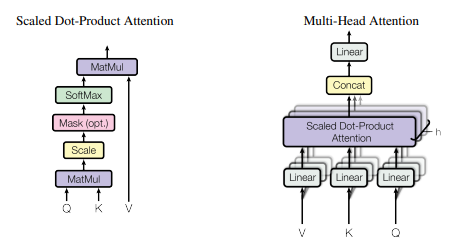


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.그림 2: (왼쪽) Scaled Dot-Product Attention. (오른쪽) Multi-Head Attention은 병렬로 실행되는 여러 주의 레이어로 구성됩니다.

**3.2.1 Scaled Dot-Product Attention**

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension , and values of dimension . We compute the dot products of the query with all keys, divide each by , and apply a softmax function to obtain the weights on the values.우리는 우리의 특별한 주의를 "Scaled Dot-Product Attention"(그림 2)이라고 부릅니다. 입력은 차원 의 쿼리 및 키와 차원 의 값으로 구성됩니다. 모든 키로 쿼리의 내적을 계산하고, 각각을 로 나누고, 값에 대한 가중치를 얻기 위해 softmax 함수를 적용합니다.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix . The keys and values are also packed together into matrices and . We compute the matrix of outputs as:

실제로, 쿼리 세트에 대한 주의 함수를 동시에 계산하여 행렬 Q로 묶습니다. 키와 값도 행렬 K와 V로 묶입니다. 출력 행렬을 다음과 같이 계산합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of . Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

가장 일반적으로 사용되는 두 가지 주의 기능은 가산 주의[2]와 내적(승법) 주의입니다. 내적 주의는 의 스케일링 인자를 제외하고 우리 알고리즘과 동일합니다. 가산 주의는 단일 은닉층이 있는 피드포워드 네트워크를 사용하여 호환성 함수를 계산합니다. 둘은 이론적 복잡성에서 유사하지만, 내적 주의는 고도로 최적화된 행렬 곱셈 코드를 사용하여 구현할 수 있기 때문에 실제로 훨씬 빠르고 공간 효율적입니다.

While for small values of the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of [3]. We suspect that for large values of , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small . To counteract this effect, we scale the dot products by .

의 작은 값에 대해 두 메커니즘이 유사하게 수행되는 반면, 가산적 주의는 의 더 큰 값에 대한 스케일링 없이 내적 주의를 능가합니다[3]. 의 큰 값에 대해 내적은 크기가 크게 증가하여 softmax 함수를 매우 작은 기울기^4가 있는 영역으로 밀어넣는다고 생각합니다. 이 효과를 상쇄하기 위해 내적을 로 스케일링합니다.

**3.2.2 Multi-Head Attention**

Instead of performing a single attention function with -dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding -dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

차원 키, 값 및 쿼리를 사용하여 단일 주의 기능을 수행하는 대신 각각 , 및 차원에 대해 학습된 서로 다른 선형 프로젝션을 사용하여 쿼리, 키 및 값을 h번 선형으로 프로젝션하는 것이 유익함을 발견했습니다. 이러한 각 예상 버전의 쿼리, 키 및 값에 대해 주의 기능을 병렬로 수행하여 -차원 출력 값을 생성합니다. 이것들은 연결되고 다시 한 번 투영되어 그림 2와 같이 최종 값이 됩니다.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.다중 헤드 주의를 통해 모델은 서로 다른 위치에 있는 서로 다른 표현 부분 공간의 정보에 공동으로 주의를 기울일 수 있습니다. 단일 주의 헤드의 경우 평균화는 이를 억제합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Where the projections are parameter matrices .

여기서 투영은 매개변수 행렬 .

In this work we employ h = 8 parallel attention layers, or heads. For each of these we use . Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.이 작업에서 우리는 h = 8개의 병렬 주의 레이어 또는 헤드를 사용합니다. 이들 각각에 대해 를 사용합니다. 각 헤드의 축소된 차원으로 인해 총 계산 비용은 전체 차원을 가진 단일 헤드 주의 비용과 유사합니다.

**3.2.3 Applications of Attention in our Model**

The Transformer uses multi-head attention in three different ways:Transformer는 다음과 같은 세 가지 방식으로 다중 헤드 어텐션을 사용합니다.

• In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].

• "인코더-디코더 주의" 계층에서 쿼리는 이전 디코더 계층에서 가져오고 메모리 키와 값은 인코더 출력에서 ​​가져옵니다. 이를 통해 디코더의 모든 위치가 입력 시퀀스의 모든 위치를 처리할 수 있습니다. 이는 [38, 2, 9]와 같은 sequence-to-sequence 모델에서 일반적인 인코더-디코더 주의 메커니즘을 모방합니다.

• The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.

인코더에는 자체 주의 레이어가 포함되어 있습니다. self-attention layer에서 모든 키, 값 및 쿼리는 같은 위치에서 옵니다. 이 경우 인코더에서 이전 레이어의 출력입니다. 인코더의 각 위치는 인코더의 이전 계층에 있는 모든 위치를 처리할 수 있습니다.

• Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to −∞) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

• 유사하게, 디코더의 self-attention 레이어는 디코더의 각 위치가 디코더의 해당 위치까지 그리고 그 위치를 포함하는 모든 위치에 주의를 기울일 수 있도록 합니다. 자동 회귀 속성을 유지하려면 디코더에서 왼쪽으로 정보 흐름을 방지해야 합니다. 잘못된 연결에 해당하는 softmax 입력의 모든 값을 마스킹(-∞로 설정)하여 scaled dot-product Attention 내부에서 이것을 구현합니다. 그림 2를 참조하십시오.

**3.3 Position-wise Feed-Forward Networks**

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.주의 하위 계층 외에도 인코더 및 디코더의 각 계층에는 각 위치에 개별적으로 동일하게 적용되는 완전히 연결된 피드포워드 네트워크가 포함됩니다. 이것은 사이에 ReLU 활성화가 있는 두 개의 선형 변환으로 구성됩니다.



While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is = 512, and the inner-layer has dimensionality = 2048.선형 변환은 다른 위치에서 동일하지만 레이어마다 다른 매개변수를 사용합니다. 이것을 설명하는 또 다른 방법은 커널 크기가 1인 두 개의 컨볼루션입니다. 입력 및 출력의 차원은 = 512이고 내부 계층의 차원은 = 2048입니다.

**3.4 Embeddings and Softmax**

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension . We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by .

다른 시퀀스 변환 모델과 마찬가지로 학습된 임베딩을 사용하여 입력 토큰과 출력 토큰을 차원의 벡터로 변환합니다. 우리는 또한 일반적인 학습된 선형 변환 및 softmax 함수를 사용하여 디코더 출력을 예측된 다음 토큰 확률로 변환합니다. 우리 모델에서 우리는 [30]과 유사하게 두 개의 임베딩 레이어와 pre-softmax 선형 변환 간에 동일한 가중치 행렬을 공유합니다. 임베딩 레이어에서 이러한 가중치를 로 곱합니다.

**3.5 Positional Encoding**

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

우리 모델에는 반복과 회선이 포함되어 있지 않기 때문에 모델이 시퀀스의 순서를 사용하려면 시퀀스에서 토큰의 상대 또는 절대 위치에 대한 정보를 주입해야 합니다. 이를 위해 인코더 및 디코더 스택의 맨 아래에 있는 입력 임베딩에 "위치 인코딩"을 추가합니다. 위치 인코딩은 임베딩과 동일한 차원 d\_model을 가지므로 둘을 합산할 수 있습니다. 학습되고 고정된 위치 인코딩에는 많은 선택이 있습니다[9].

텍스트이(가) 표시된 사진

자동 생성된 설명

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. is the sequence length, is the representation dimension, is the kernel size of convolutions and the size of the neighborhood in restricted self-attention.표 1: 다양한 계층 유형에 대한 최대 경로 길이, 계층당 복잡성 및 최소 순차 작업 수. n은 시퀀스 길이, d는 표현 차원, k는 컨볼루션의 커널 크기, r은 제한된 자가 주의에서 이웃 크기입니다.

In this work, we use sine and cosine functions of different frequencies:이 작업에서는 서로 다른 주파수의 사인 및 코사인 함수를 사용합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

where is the position and is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from to 10000 · . We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, can be represented as a linear function of .여기서 pos는 위치이고 는 차원입니다. 즉, 위치 인코딩의 각 차원은 정현파에 해당합니다. 파장은 에서 10000 · 까지 기하학적 진행을 형성합니다. 고정 오프셋 k에 대해 가 의 선형 함수로 표현될 수 있기 때문에 모델이 상대 위치에 의해 참석하는 법을 쉽게 학습할 수 있다고 가정했기 때문에 이 함수를 선택했습니다.

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

우리는 또한 학습된 위치 임베딩[9]을 대신 사용하여 실험했으며 두 버전이 거의 동일한 결과를 생성한다는 것을 발견했습니다(표 3 행(E) 참조). 모델이 훈련 중에 발생한 것보다 긴 시퀀스 길이로 외삽할 수 있기 때문에 사인파 버전을 선택했습니다.

**4 Why Self-Attention**

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations to another sequence of equal length , with , such as a hiddenlayer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.이 섹션에서는 self-attention 레이어의 다양한 측면을 하나의 가변 길이 심볼 표현 시퀀스를 동일한 길이의 다른 시퀀스에 매핑하는 데 일반적으로 사용되는 순환 및 컨볼루션 레이어와 비교합니다. , 예를 들어 일반적인 시퀀스 변환 인코더 또는 디코더의 은닉층입니다. 자기 주의를 사용하도록 동기를 부여하기 위해 우리는 세 가지 열망을 고려합니다.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.하나는 계층당 총 계산 복잡도입니다. 또 다른 하나는 필요한 순차 작업의 최소 수로 측정한 병렬 처리할 수 있는 계산의 양입니다.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

세 번째는 네트워크의 장거리 종속성 간의 경로 길이입니다. 장거리 종속성을 학습하는 것은 많은 시퀀스 변환 작업에서 핵심 과제입니다. 이러한 종속성을 학습하는 능력에 영향을 미치는 한 가지 핵심 요소는 네트워크에서 순방향 및 역방향 신호가 통과해야 하는 경로의 길이입니다. 입력 시퀀스와 출력 시퀀스의 위치 조합 사이의 경로가 짧을수록 장거리 종속성을 더 쉽게 학습할 수 있습니다. 따라서 다른 계층 유형으로 구성된 네트워크에서 두 입력 및 출력 위치 사이의 최대 경로 길이도 비교합니다.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence length is smaller than the representation dimensionality d, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [38] and byte-pair [31] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position. This would increase the maximum path length to . We plan to investigate this approach further in future work.

표 1에서 알 수 있듯이 self-attention layer는 모든 위치를 일정한 수의 순차적으로 실행되는 작업으로 연결하는 반면, recurrent layer는 O(n)개의 순차적 작업을 필요로 합니다. 계산 복잡성의 관점에서, self-attention layer는 시퀀스 길이 n이 표현 차원 d보다 작을 때 순환 레이어보다 빠릅니다. 이는 기계 번역의 최첨단 모델에서 사용되는 문장 표현의 경우 가장 자주 발생합니다. , 단어 조각 [38] 및 바이트 쌍 [31] 표현과 같은. 매우 긴 시퀀스를 포함하는 작업에 대한 계산 성능을 개선하기 위해 self-attention은 각 출력 위치를 중심으로 하는 입력 시퀀스에서 크기 r의 이웃만 고려하도록 제한될 수 있습니다. 이렇게 하면 최대 경로 길이가 O(n/r)로 늘어납니다. 향후 작업에서 이 접근 방식을 더 조사할 계획입니다.

A single convolutional layer with kernel width does not connect all pairs of input and output positions. Doing so requires a stack of ) convolutional layers in the case of contiguous kernels, or in the case of dilated convolutions [18], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of k. Separable convolutions [6], however, decrease the complexity considerably, to . Even with , however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

커널 너비가 k<n인 단일 컨볼루션 레이어는 모든 입력 및 출력 위치 쌍을 연결하지 않습니다. 그렇게 하려면 연속 커널의 경우 O(n/k) 개의 컨볼루션 레이어 스택이 필요하고, 확장된 컨볼루션의 경우 [18], 두 레이어 사이에서 가장 긴 경로의 길이가 증가합니다. 네트워크에서의 위치. 컨볼루션 레이어는 일반적으로 순환 레이어보다 k 배 더 비쌉니다. 그러나 분리 가능한 컨볼루션[6]은 로 복잡성을 상당히 줄입니다. 그러나 k=n인 경우에도 분리 가능한 컨볼루션의 복잡성은 self-attention layer와 point-wise feed-forward layer의 조합과 동일하며, 이는 우리가 모델에서 취하는 접근 방식입니다.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

부수적인 이점으로 자기 주의는 더 해석 가능한 모델을 생성할 수 있습니다. 우리는 모델의 주의 분포를 검사하고 부록에서 예제를 제시하고 논의합니다. 개별 주의 머리는 다른 작업을 수행하는 방법을 분명히 배울 뿐만 아니라 많은 사람들이 문장의 구문 및 의미 구조와 관련된 행동을 보이는 것으로 보입니다.

**5 Training**

This section describes the training regime for our models.

이 섹션에서는 우리 모델의 훈련 체제를 설명합니다.

**5.1 Training Data and Batching**

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared sourcetarget vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

우리는 약 450만 문장 쌍으로 구성된 표준 WMT 2014 영어-독일어 데이터 세트에 대해 교육했습니다. 문장은 약 37000 토큰의 공유 소스 대상 어휘가 있는 바이트 쌍 인코딩[3]을 사용하여 인코딩되었습니다. 영어-프랑스어의 경우 36M 문장과 토큰을 32000단어 단어로 분할하여 구성된 훨씬 더 큰 WMT 2014 영어-프랑스어 데이터 세트를 사용했습니다[38]. 문장 쌍은 대략적인 시퀀스 길이로 함께 배치되었습니다. 각 훈련 배치에는 약 25000개의 소스 토큰과 25000개의 대상 토큰이 포함된 문장 쌍 세트가 포함되어 있습니다.

**5.2 Hardware and Schedule**

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

우리는 8개의 NVIDIA P100 GPU가 있는 한 머신에서 모델을 훈련했습니다. 논문 전체에 설명된 하이퍼파라미터를 사용하는 기본 모델의 경우 각 훈련 단계는 약 0.4초가 소요되었습니다. 우리는 총 100,000걸음 또는 12시간 동안 기본 모델을 훈련했습니다. 우리의 큰 모델의 경우(표 3의 맨 아래 줄에 설명됨) 단계 시간은 1.0초였습니다. 큰 모델은 30만 단계(3.5일) 동안 훈련되었습니다.

**5.3 Optimizer**

We used the Adam optimizer [20] with . We varied the learning rate over the course of training, according to the formula:

우리는 및 인 Adam 옵티마이저를 사용했습니다. 다음 공식에 따라 교육 과정에서 학습률을 변경했습니다.



This corresponds to increasing the learning rate linearly for the first warmup\_steps training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used warmup\_steps = 4000.이것은 첫 번째 warmup\_steps 훈련 단계에 대해 선형적으로 학습률을 증가시키고 이후에는 단계 수의 역제곱근에 비례하여 감소시키는 것에 해당합니다. 우리는 warmup\_steps = 4000을 사용했습니다.

**5.4 Regularization**

We employ three types of regularization during training:훈련 중에 세 가지 유형의 정규화를 사용합니다.

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of .

Residual Dropout 우리는 dropout[33]을 sub-layer 입력에 추가되고 정규화되기 전에 각 sub-layer의 출력에 적용합니다. 또한 인코더 및 디코더 스택 모두에서 임베딩 및 위치 인코딩의 합에 드롭아웃을 적용합니다. 기본 모델의 경우 비율을 사용합니다.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

표 2: Transformer는 훨씬 적은 교육 비용으로 English-to-German 및 English-to-French newstest2014 테스트에서 이전의 최첨단 모델보다 더 나은 BLEU 점수를 달성했습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

Label Smoothing During training, we employed label smoothing of value . This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

레이블 평활화 훈련 중에 ϵ\_ls=0.1 값의 레이블 평활화를 사용했습니다. 이는 모델이 더 불확실한 것을 학습하므로 혼란을 야기하지만 정확도와 BLEU 점수를 향상시킵니다.

**6 Results**

**6.1 Machine Translation**

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

WMT 2014 영어-독일어 번역 작업에서 빅 트랜스포머 모델(표 2의 트랜스포머(큰))은 이전에 보고된 최고의 모델(앙상블 포함)보다 2.0 BLEU 이상 성능이 뛰어나 새로운 상태를 확립합니다. 아트 BLEU 점수 28.4. 이 모델의 구성은 표 3의 맨 아래 줄에 나열되어 있습니다. 교육은 8개의 P100 GPU에서 3.5일이 걸렸습니다. 우리의 기본 모델조차도 경쟁 모델의 훈련 비용의 일부만으로 이전에 발표된 모든 모델과 앙상블을 능가합니다.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0,outperforming all of the previously published single models, at less than 1/4 the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate Pdrop = 0.1, instead of 0.3.

WMT 2014 영어-프랑스어 번역 작업에서 우리의 빅 모델은 41.0의 BLEU 점수를 달성하여 이전 최신 기술의 교육 비용의 1/4 미만으로 이전에 게시된 모든 단일 모델을 능가합니다. 모델. English-to-French에 대해 훈련된 Transformer(대형) 모델은 탈락률 Pdrop = 0.3 대신 0.1을 사용했습니다.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty α = 0.6 [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38].

기본 모델의 경우 10분 간격으로 작성된 마지막 5개의 체크포인트를 평균하여 얻은 단일 모델을 사용했습니다. 큰 모델의 경우 마지막 20개의 체크포인트를 평균화했습니다. 우리는 빔 크기가 4이고 길이 페널티 α = 0.6인 빔 탐색을 사용했습니다[38]. 이 하이퍼파라미터는 개발 세트에 대한 실험 후에 선택되었습니다. 추론 시 최대 출력 길이를 입력 길이 + 50으로 설정하지만 가능하면 일찍 종료한다[38].

Table 2 summarizes our results and compares our translation quality and training costs to other model rchitectures from the literature. We estimate the number of floating point operations used to train a odel by multiplying the training time, the number of GPUs used, and an estimate of the sustained ingle-precision floating-point capacity of each .

표 2는 결과를 요약하고 번역 품질 및 교육 비용을 문헌의 다른 모델 아키텍처와 비교합니다. 훈련 시간, 사용된 GPU 수, 각 GPU의 유지되는 단정밀도 부동 소수점 용량 추정치를 곱하여 odel을 훈련하는 데 사용되는 부동 소수점 연산 수를 추정합니다^5

**6.2 Model Variations**

To evaluate the importance of different components of the Transformer, we varied our base model n different ways, measuring the change in performance on English-to-German translation on the development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

Transformer의 다양한 구성 요소의 중요성을 평가하기 위해 우리는 기본 모델을 다양한 방식으로 변경하여 개발 세트인 newstest2013에서 영어-독일어 번역 성능 변화를 측정했습니다. 이전 섹션에서 설명한 대로 빔 검색을 사용했지만 체크포인트 평균은 사용하지 않았습니다. 이러한 결과를 표 3에 제시합니다.

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

표 3 행(A)에서 섹션 3.2.2에 설명된 대로 계산량을 일정하게 유지하면서 주의 헤드 수와 주의 키 및 값 차원을 변경합니다. 단일 헤드 주의는 최상의 설정보다 0.9 BLEU 좋지 않지만 헤드가 너무 많으면 품질도 떨어집니다.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

표 3: Transformer 아키텍처의 변형. 목록에 없는 값은 기본 모델의 값과 동일합니다. 모든 메트릭은 영어-독일어 번역 개발 세트, newstest2013에 있습니다. 나열된 혼란은 바이트 쌍 인코딩에 따라 단어 단위이며 단어당 혼란과 비교해서는 안됩니다.

테이블이(가) 표시된 사진

자동 생성된 설명

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

표 4: Transformer는 영어 구성 요소 구문 분석에 잘 일반화됩니다(결과는 WSJ의 섹션 23에 있음).

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

In Table 3 rows (B), we observe that reducing the attention key size dk hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly identical results to the base model.

표 3의 행(B)에서 주의 키 크기 dk를 줄이면 모델 품질이 저하됨을 알 수 있습니다. 이는 호환성을 판단하는 것이 쉽지 않으며 내적보다 정교한 호환성 함수가 유리할 수 있음을 시사합니다. 우리는 행 (C)와 (D)에서 예상대로 더 큰 모델이 더 좋고 드롭아웃이 과적합을 피하는 데 매우 도움이 된다는 것을 관찰했습니다. 행 (E)에서 우리는 정현파 위치 인코딩을 학습된 위치 임베딩으로 교체하고 기본 모델과 거의 동일한 결과를 관찰합니다.

**6.3 English Constituency Parsing**

To evaluate if the Transformer can generalize to other tasks we performed experiments on English onstituency parsing. This task presents specific challenges: the output is subject to strong structural 9 constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

Transformer가 다른 작업으로 일반화할 수 있는지 평가하기 위해 영어 위치 구문 분석에 대한 실험을 수행했습니다. 이 작업은 특정 과제를 제시합니다. 출력은 강력한 구조적 제약 조건을 따르고 입력보다 훨씬 더 깁니다. 더욱이, RNN sequence-to-sequence 모델은 소규모 데이터 체제에서 최신 결과를 얻을 수 없었습니다.

We trained a 4-layer transformer with on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

Penn Treebank[25]의 월스트리트 저널(WSJ) 부분에서 d\_model=1024인 4계층 변환기를 약 40K 훈련 문장으로 훈련했습니다. 우리는 또한 약 1,700만 문장[37]에서 더 큰 높은 신뢰도와 BerkleyParser corpora를 사용하여 준지도 환경에서 훈련했습니다. 우리는 WSJ 전용 설정에 대해 16K 토큰의 어휘를 사용하고 반 감독 설정에 대해 32K 토큰의 어휘를 사용했습니다.

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. During inference, we increased the maximum output length to input length + 300. We used a beam size of 21 and α = 0.3 for both WSJ only and the semi-supervised setting.섹션 22 개발 세트에서 드롭아웃, 주의 및 잔차(섹션 5.4), 학습률 및 빔 크기를 선택하기 위해 소수의 실험만 수행했으며 다른 모든 매개변수는 영어-독일어 기본 번역 모델에서 변경되지 않은 상태로 유지되었습니다. 추론하는 동안 최대 출력 길이를 입력 길이 + 300으로 늘렸습니다. WSJ 전용 및 반 감독 설정 모두에 대해 빔 크기 21 및 α = 0.3을 사용했습니다.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8].

표 4의 결과는 작업별 조정이 없음에도 불구하고 우리 모델이 놀라울 정도로 잘 수행되어 Recurrent Neural Network Grammar[8]를 제외하고 이전에 보고된 모든 모델보다 더 나은 결과를 산출함을 보여줍니다.

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the BerkeleyParser [29] even when training only on the WSJ training set of 40K sentences.

RNN sequence-to-sequence 모델[37]과 달리 Transformer는 40K 문장의 WSJ 훈련 세트에서만 훈련하는 경우에도 BerkeleyParser[29]보다 성능이 뛰어납니다.

**7 Conclusion**

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

이 작업에서 우리는 인코더-디코더 아키텍처에서 가장 일반적으로 사용되는 순환 레이어를 다중 헤드 셀프 어텐션으로 대체하는 완전히 어텐션에 기반한 최초의 시퀀스 변환 모델인 Transformer를 제시했습니다.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

번역 작업의 경우 Transformer는 반복 또는 컨볼루션 계층을 기반으로 하는 아키텍처보다 훨씬 빠르게 훈련할 수 있습니다. WMT 2014 영어-독일어 및 WMT 2014 영어-프랑스어 번역 작업에서 우리는 새로운 기술 수준을 달성했습니다. 이전 작업에서 우리의 최상의 모델은 이전에 보고된 모든 앙상블보다 성능이 뛰어납니다.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours. The code we used to train and evaluate our models is available at https://github.com/ tensorflow/tensor2tensor.

우리는 주의 기반 모델의 미래에 대해 기대하고 있으며 이를 다른 작업에 적용할 계획입니다. 우리는 Transformer를 텍스트 이외의 입력 및 출력 양식과 관련된 문제로 확장하고 이미지, 오디오 및 비디오와 같은 대규모 입력 및 출력을 효율적으로 처리하기 위해 로컬의 제한된 주의 메커니즘을 조사할 계획입니다. 세대를 덜 순차적으로 만드는 것은 우리의 또 다른 연구 목표입니다. 모델을 훈련하고 평가하는 데 사용한 코드는 https://github.com/tensorflow/tensor2tensor에서 사용할 수 있습니다.

Acknowledgements We are grateful to Nal Kalchbrenner and Stephan Gouws for their fruitful comments, corrections and inspiration.

감사의 말 우리는 유익한 논평, 수정 및 영감을 준 Nal Kalchbrenner와 Stephan Gouws에게 감사드립니다.